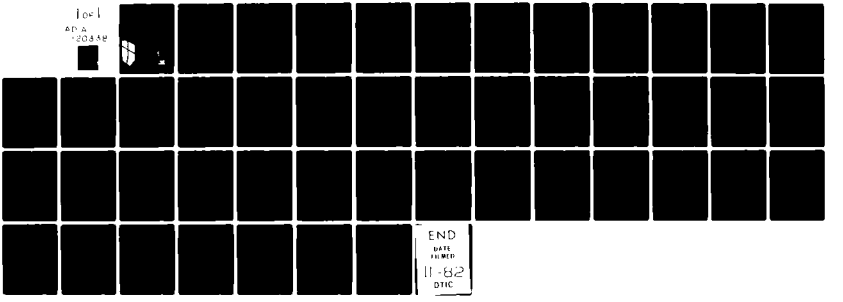
AD-A120 338 CONSTRUCTION ENGINEERING RESEARCH LAB (ARMY) CHAMPAIGN IL F/G 9/2
USER'S MANUAL FOR MILENG1/UTIL READ-ONLY MEMORY MODULE OF THE C--ETC(U)
SEP 82 J M DEPONAI
UNCLASSIFIED CERL-TR-P-136 NL



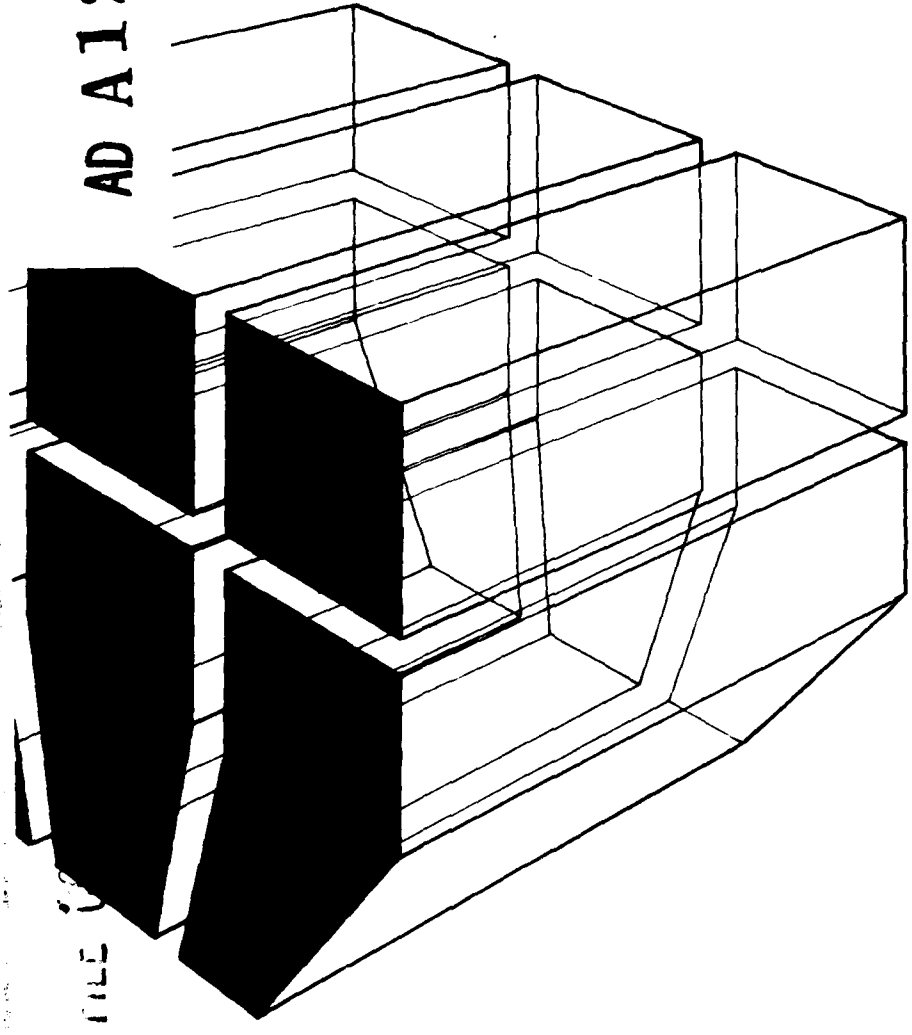
END
DATE FILMED
11-82
DTIC

construction engineering research laboratory

United States Army
Corps of Engineers
...Serving the Army
...Serving the Nation

TECHNICAL REPORT P-136
September 1982

USER'S MANUAL FOR MILENG1/UTIL READ-ONLY-MEMORY MODULE OF THE COMBAT ENGINEER PROGRAMMABLE HAND-HELD CALCULATOR



by
John M. Deponai III

CERL

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CERL-TR-P-136	2. GOVT ACCESSION NO. AD-A120 338	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) USER'S MANUAL FOR MILENG1/UTIL READ ONLY MEMORY MODULE OF THE COMBAT ENGINEER PROGRAMMABLE HAND-HELD CALCULATOR		5. TYPE OF REPORT & PERIOD COVERED FINAL 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John M. Deponai III		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. ARMY CONSTRUCTION ENGINEERING RESEARCH LABORATORY P.O. BOX 4005, CHAMPAIGN, IL 61820		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 4A762731AT41-D-049
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE September 1982 13. NUMBER OF PAGES 45
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Copies are obtainable from the National Technical Information Service
Springfield, VA 22151

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

military engineering
calculators
programmable calculators
HP-41C

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Six pilot combat engineering programs and ten programming utility subroutines have been developed as part of an ongoing study to evaluate whether programmable hand-held calculators can increase the efficiency or military effectiveness of engineer troop units from platoon through brigade levels. This report describes the programs, explains how to use them, and gives example problems.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

This investigation was conducted for the Directorate of Military Programs, Office of the Chief of Engineers (OCE), under Project 4A762731AT41, "Design, Construction, and Operation and Maintenance Technology for Military Facilities"; Task D, "Combat Engineering Strategy"; Work Unit 049, "Programmable Calculator Technology for Engineer Troop Units." The applicable STO is 81-5.1:19. The OCE Technical Monitors were LTC John Howard and Dr. Clemens Meyer, both of DAEN-ZCM.

This work was performed by the Facility Systems Division (FS) of the U.S. Army Construction Engineering Research Laboratory (CERL). Mr. E. A. Lotz is Chief of CERL-FS.

The cooperation and contributions of many persons on the staff of the U.S. Army Engineer School are gratefully acknowledged. CPT Scott Loomer of the Defense Mapping School is especially commended for developing the Bridge Classification Program and all but two of the global utility routines described in this report.

COL Louis J. Circeo is Commander and Director of CERL, and Dr. L. R. Shaffer is Technical Director.

CONTENTS

TABLES

FIGURES

FIGURES (Cont'd)

USER'S MANUAL FOR MILENG1/UTIL READ ONLY MEMORY MODULE OF THE COMBAT ENGINEER PROGRAMMABLE HAND-HELD CALCULATOR

1 INTRODUCTION

Background

Recent advancements in the state of the art of programmable calculators have indicated that these devices might help military engineers work more efficiently. To determine the potential of these systems, in March 1980 the U.S. Army Engineer School asked the U.S. Army Construction Engineering Research Laboratory (CERL) to see how hand-held programmable calculators could be exploited by combat engineers. To date, six pilot programs and ten utility routines have been developed for testing. Details of the study and comprehensive information on the programs and routines appear in CERL Technical Report P-134, Software Documentation for MILENG1/UTIL Read Only Memory Module.

This user's manual describes each MILENG1/UTIL program, explains how to use each program, and gives example problems.

Conventions

Only a few conventions must be learned to use the MILENG1/UTIL programs effectively.

Yes/No Input

When the program asks a question that needs a "yes" or "no" answer, the calculator will display an alpha string ending in (Y/N)?. To respond "yes", press two keys: Y and R/S.* To answer "no", press the N and R/S keys. Any other responses will be rejected by the program and the question will be displayed again. When the program asks this type of question, it automatically puts the calculator in the alpha mode, then awaits your response.

Numeric Input

When the program asks for numeric input the calculator displays an alpha string concatenated with a unit of measurement and an =?. To respond, key in a numeric string, then press the R/S key. (Do not use the ENTER key.) If an alpha string were keyed in, it would be rejected and the question would be repeated. If the input is not within the allowable range specified in the program, one of the following messages will be displayed:

---

* When several letters/symbols are underlined, it means that as a group they identify the name of a single key on the calculator. Only key function names will be shown. Use of the shift key, if needed, is assumed. Single letters or numbers represent individual keys on the calculator.

MUST BE <= (some maximum value)

or

MUST BE >= (some minimum value).

Then, the original question will be repeated.

Output

All "MILENG1/UTIL" programs can run with or without a printer. If a printer is attached, the program stops only (1) when your input is required and (2) at the end of a program. If a printer is not attached, the program also will stop after each line of output. To continue execution, press the R/S key each time the program stops. If you use a printer, set it to the NORM mode.

Program Access

To access a particular program, you must "execute" the program name. For example, to call the MINES program, press XEQ ALPHAMINESALPHA and the program will begin execution.

Each program may be assigned to almost any key on the HP-41 calculator. For example, to assign the MINES program to the $\sqrt{x}$ key, press ASN ALPHAMINESALPHA $\sqrt{x}$. Then, when the calculator is in the USER mode, the MINES program will be executed if the $\sqrt{x}$ key is pressed. See the HP-41 Owner's Handbook and Programming Guide for more information on this "assign" feature.

Size Check

Before a program can be executed there must be enough data registers available to run the program. The minimum number of registers required for each program is:

Program	Registers Required
BRDGCLS	52
CPM	*(2A+43)
CRATER	40
DEMO	41
MINES	53
WIRE	44

*A = Number of Activity Nodes

To set a program to the correct size, press XEQ ALPHASIZEALPHA. The calculator then will prompt for the number of registers required. A three-digit number must be entered; i.e., for the BRDGCLS program, enter 052. If the program is sized incorrectly, the program will immediately tell you to

RESIZE > (No. of Registers Required, minus one).

For example, if the BRDGCLS program was originally sized at 040, the BRDCGLS program would display the message RESIZE> 51. You would then press XEQ ALPHASIZEALPHA 052 (or greater), and then execute the program name again.

Register Use

Registers 00 through 19 are reserved for your use. These registers are not used by any of the programs on MILENG1/UTIL. However, the contents of registers 20 and above are affected, depending on which programs are executed.

2 THE BRIDGE CLASSIFICATION PROGRAM (BRDGCLS)

The bridge classification program is used with certain tables in FM 5-34 (September 1976) to help you determine bridge superstructure classification. This program may be used for both timber and steel stringer bridges. It first asks you for the basic dimensions of the bridge. At the appropriate times, it refers you to certain tables and figures in FM 5-34, tells you what entry values are needed, and asks for the value of the variables corresponding to those entry conditions. You extract the appropriate value from the manual's table (or figure) and input it to the calculator. The program determines the limiting classifications for one- and two-way traffic by wheeled and tracked vehicles. It determines the constraints imposed by moment capacity, shear capacity, deck thickness, roadway width, and also determines if additional braces are required.

General Program Information

Abbreviations used in BRDGCLS are listed in Table 1. Input variable operating limits are listed in Table 2.

Program Sequence

The typical sequence of events and the options you encounter when executing this program are shown in Figure 1.

BRDGCLS Program Example

Assume that a bridge reconnaisance was conducted to obtain the information below for a timber trestle bridge. Determine the final bridge classification.

Road Width = 23 ft
Nine Timber Stringers, each 8" x 18"
Deck = two layers of 3" x 12" plank
Two Lateral Braces, 8" x 10" at midpoint and at one end of span.
Span Length = 17 ft
Springer Spacing = 35 in.

Figure 2 shows how to solve this problem using the BRGCLS program. Figure 3 is an example of BRGCLS output with printer attached.

Table 1

BRDGCLS Abbreviations

Symbol	Meaning
CLS,CLASS	Classification
DT	Deck Thickness
FIG	Figure
FT	Feet
IN	Inches
KP	Kip
L,M	Maximum Span Length
LAM	Laminated
LN	Lane
M	Moment Capacity
M,DL	Dead Load Moment
M,LL	Live Load Moment
N1	Effective Number of Stringers/Lane
N2	Effective Number of Stringers/Lane for a 2-Lane Bridge
RECON	Reconnaissance
S,B	Maximum Bracing Spacing
S,S	Stringer Spacing
STL	Steel
STR	Stringer
TAB	Table
TBR	Timber
THICK	Thickness
V	Shear Capacity
V,DL	Dead Load Shear
V,LL	Live Load Shear
WY	Way
(Y/N)	(Yes/No)
#	Number
%	Percent

Table 2

BRDGCLS Program Input Variable Operating Limits

Variable	Units	Minimum	Maximum
Road width	Feet	8	50
Span length	Feet	10	200
Stringer number	Each	2	25
Stringer width	Inches	4	20
Stringer depth	Inches	6	60
Stringer flange thickness	Inches	.3	2
Stringer spacing	Inches	10	100
Deck Thickness	Inches	2	12
% laminated	%	0	100
Number of braces	Each	0	20
Moment capacity	Kip-Feet	3	3100
Shear capacity	Kip	3	600
Maximum span length	Feet	9	135
Maximum bracing spacing	Feet	6	26
Dead load moment	Kip-Feet	3	1200
Dead load shear	Kip	1	65
Wheel classification	Class	0	150
Track classification	Class	0	150
Classification	Class	0	150

Source Reference Notes:

FM 5-34 (September 1976), pps 170-184 and 199-203.

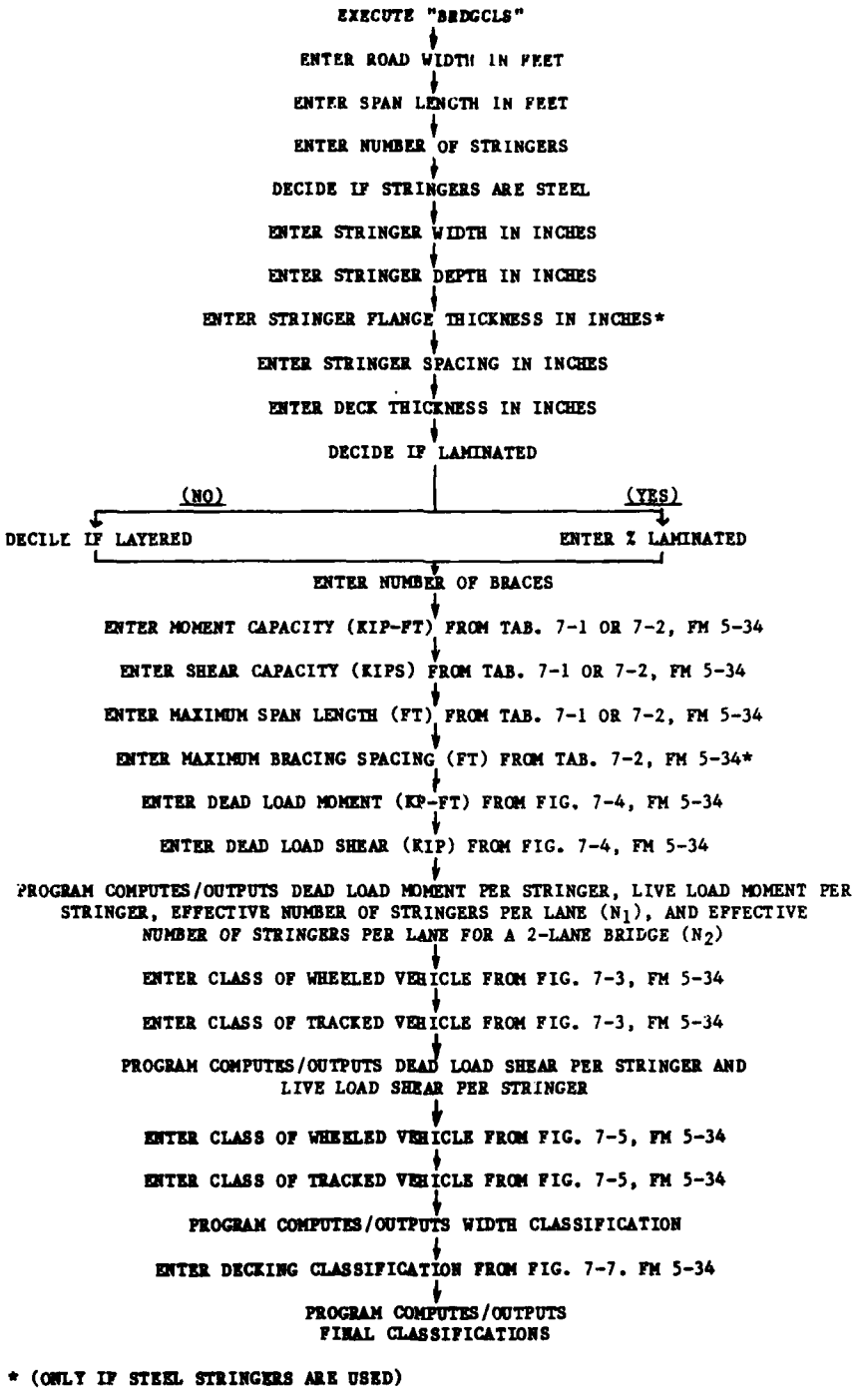


Figure 1. BRDGCLS program sequence.

Step	Press	Resulting Display	Comments
1	XEQ		
	ALPHA		
	BRDGCLS		
	ALPHA	BRIDGE CLASS	
2	R/S	RECON:	
3	R/S	ROAD WIDTH (FT) = ?	
4	23 R/S	SPAN LENGTH (FT) = ?	
5	17 R/S	STRINGERS:	
6	R/S	STR, NUMBER = ?	
7	9 R/S	STEEL (Y/N)?	
8	N R/S	STR. WIDTH (IN) = ?	
9	8 R/S	STR. DEPTH (IN) = ?	
10	18 R/S	STR. SPACING (IN) = ?	
11	35 R/S	DECK THICK. (IN) = ?	
12	6 R/S	LAMINATED (Y/N)?	
13	N R/S	DECK LAYERED (Y/N)?	
14	Y R/S	# BRACES = ?	
15	2 R/S	CLASS:	
16	R/S	TAB. 7-1, FM 5-34:	See Note 1
17	R/S	8.0X18.0	
18	R/S	M(KP-FT) = ?	
19	86.40 R/S	V (KIP) = ?	
20	14.40 R/S	L,M (FT) = ?	
21	21.50 R/S	FIG. 7-4, FM 5-34:	See Note 2
22	R/S	TBR, 2 LN, 17. FT:	
23	R/S	M,DL (KP-FT) = ?	
24	37.36 R/S	V,DL (KIP) = ?	
25	8.75 R/S	M,DL/STR= 4.2	
26	R/S	M,LL/STR=82.2	
27	R/S	N1 = 2.71	
28	R/S	N2 = 3.38	
29	R/S	FIG. 7-3, FM 5-34:	See Note 3
30	R/S	17 FT. & 223. M,LL:	
31	R/S	CLS, WHEEL = ?	
32	60 R/S	CLS, TRACK = ?	
33	40 R/S	V,DL/STR = 1.0	
34	R/S	V,LL/STR = 13.4	
35	R/S	FIG. 7-5, FM 5-34:	See Note 4
36	R/S	17 FT. & 52. V,LL:	
37	R/S	CLS, WHEEL = ?	
38	50 R/S	CLS, TRACK = ?	
39	40 R/S	WIDTH CLS:	
40	R/S	1 WAY = 100	
41	R/S	2 WAY = 30	
42	R/S	FIG. 7-7, FM 5-34:	See Note 5
43	R/S	DT = 4.0 & S,S = 35.:	
44	R/S	CLASS = ?	

Figure 2. BRDGCLS program problem example.

Step	Press	Resulting Display	Comments
45	12 R/S	FINAL CLASS:	
46	R/S	ADD 1.BRACES	See Note 6
47	R/S	1 WY WHEEL = 12.	
48	R/S	2 WY WHEEL = 12.	
49	R/S	1 WY TRACK = 12.	
50	R/S	2 WY TRACK = 12.	
51	R/S	END PROGRAM	

Notes:

1. See Table 7-1, FM 5-34 for the properties of timber stringers.

2. See Figure 7-4, FM 5-34 to determine the dead load moment and shear.

3. See Figure 7-3, FM 5-34 to determine the wheel and track classification based on moment capacity.

4. See Figure 7-5, FM 5-34 to determine the wheel and track classification based on shear capacity.

5. See Figure 7-7, FM 5-34 to determine the classification based on decking thickness.

6. If a printer is connected and in the NORM mode, steps 46 through 51 will be output automatically.

Figure 2. (Cont'd).

```
          XROM "BRDGCLS"
BRIDGE CLASS
RECON
ROAD WIDTH(FT)=?
              23.       RUN
SPAN LENGTH(FT)=?
              17.       RUN
STRINGERS:
STR, NUMBER=?
               9.       RUN
STEEL(Y/N)?
N                        RUN
STR. WIDTH(IN)=?
               8.       RUN
STR. DEPTH(IN)=?
              18.       RUN
STR. SPACING(IN)=?
              35.       RUN
DECK THICK.(IN)=?
               6.       RUN
LAMINATED(Y/N)?
N                        RUN
DECK LAYERED(Y/N)?
Y                        RUN
#BRACES=?
               2.       RUN
CLASS:
TAB.7-1,FM5-34:
8.0X81.0
M(KP-FT)=?
           86.40        RUN
V(KIP)=?
           14.40        RUN
L,M(FT)=?
           21.50        RUN
FIG.7-4,FM5-34.
TBR,2 LN,17.FT:
M,DL(KP-FT)=?
           37.36        RUN
V,DL(KIP)=?
            8.75        RUN
M,DL/STR=4.2
M,LL/STR=82.2
N1=2.71
N2=3.38
FIG.7-3,FM5-34:
17.FT.&223.M,LL:
CLS, WHEEL=?
              60.       RUN
CLS, TRACK=?
              40.       RUN
V,DL/STR=1.0
V,LL/STR=13.4
FIG.7-5,FM5-34:
17.FT.&52.V,LL:
CLS, WHEEL=?
              50.       RUN
CLS, TRACK=?
              40.       RUN
WIDTH CLS:
1 WAY=100.
2 WAY=30.
FIG.7-7,FM5-34:
DT=4.0&S,S=35.:
CLASS=?
              12.       RUN
FINAL CLASS:
ADD 1. BRACES
1WY WHEEL=12.
2WY WHEEL=12.
1WY TRACK=12.
2WY TRACK=12.
END PROGRAM
```

Figure 3. BRDGCLS program example (with printer).

3 THE CRITICAL PATH METHOD PROGRAM (CPM)

The CPM program provides an easy way to do the tedious calculations associated with using CPM project control. The CPM program uses activity-on-the-node logic. Up to 98 activities can be analyzed if the full HP-41cv resident capacity for data storage is used. Up to 20 activities can be done on the HP-41c model without memory modules.

The program computes the total float, the early start time, the early finish time, the late start time, and the late finish time for each activity. The "with printer" version prints out in boxes. These boxes can be cut out and pasted together to graph logical relationships. In the printer version, the preceding activities for each activity are noted to the left of each diagram, so the user will know how to connect the activities together. The "without printer" output must be copied as it is output. Then diagrams can be drawn by hand and annotated with the correct information.

General Program Information

Abbreviations used in the CPM program are listed in Table 3. Variable input operating limits are listed in Table 4.

Program Sequence

The typical sequence of events and the options you encounter when executing this program are shown in Figure 4.

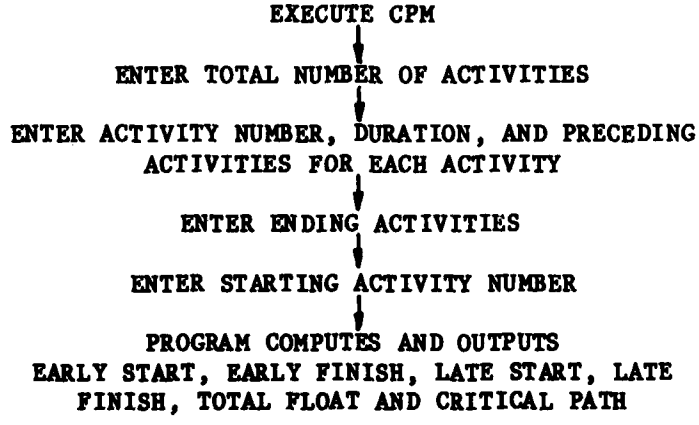


Figure 4. CPM program sequence.

Table 3

CPM Program Abbreviations

Symbol	Meaning
ACT	Activity
CPM	Critical Path Method
EF	Early Finish
END	Ending
ES	Early Start
LF	Late Finish
LS	Late Start
PRED	Preceding
START	Starting
TF	Total Float
(Y/N)	(Yes/No)
#	Number

Table 4

CPM Program Variable Input Operating Limits

Variable	Minimum	Maximum
Total Number of Activities	1	98
Activity Identification Numbers	1*	98
Duration	0	1000
Preceding Activity Identification Numbers	1*	98
Starting Activity Number	1	98

*0 is also a valid input but tells the program that there are no more inputs for these variables.

The limits set for preceding activity numbers (i.e., the identification number) show that you can enter any activity between 1 and 98. However, the total number of preceding activities for a single activity cannot exceed five. The registers will be disrupted if more than five preceding activities or more than 25 ending activities are used. The CPM program is not designed to check these last two limits.

CPM Program Example

Given the following information, determine the early start, early finish, late start, late finish, and total float for each activity using the critical path method.

Activity No.	Duration	Preceding Activity
1	15	--
2	20	1
3	18	1
4	18	3
5	20	2,4

Figure 5 shows how to solve this problem using the CPM program. Figure 6 is an example of CPM program output with printer attached.

Table 3

CPM Program Abbreviations

Symbol	Meaning
ACT	Activity
CPM	Critical Path Method
EF	Early Finish
END	Ending
ES	Early Start
LF	Late Finish
LS	Late Start
PRED	Preceding
START	Starting
TF	Total Float
(Y/N)	(Yes/No)
#	Number

Table 4

CPM Program Variable Input Operating Limits

Variable	Minimum	Maximum
Total Number of Activities	1	98
Activity Identification Numbers	1*	98
Duration	0	1000
Preceding Activity Identification Numbers	1*	98
Starting Activity Number	1	98

*0 is also a valid input but tells the program that there are no more inputs for these variables.

The limits set for preceding activity numbers (i.e., the identification number) show that you can enter any activity between 1 and 98. However, the total number of preceding activities for a single activity cannot exceed five. The registers will be disrupted if more than five preceding activities or more than 25 ending activities are used. The CPM program is not designed to check these last two limits.

Step	Press	Resulting Display	Comments
1	XEQ ALPHA CPM ALPHA	TOTAL # ACTIVITIES = ?	
2	5 R/S	ACTIVITY # = ?	See Note 1
3	1 R/S	DURATION = ?	
4	15 R/S	PRED. ACT = ?	
5	0 R/S	ACTIVITY # = ?	
6	2 R/S	DURATION = ?	
7	20 R/S	PRED. ACT = ?	
8	1 R/S	PRED. ACT = ?	
9	0 R/S	ACTIVITY # = ?	
10	3 R/S	DURATION = ?	
11	18 R/S	PRED. ACT = ?	
12	1 R/S	PRED. ACT = ?	
13	0 R/S	ACTIVITY # = ?	
14	4 R/S	DURATION = ?	
15	18 R/S	PRED. ACT = ?	
16	3 R/S	PRED. ACT = ?	
17	0 R/S	ACTIVITY # = ?	
18	5 R/S	DURATION = ?	
19	20 R/S	PRED. ACT = ?	
20	2 R/S	PRED. ACT = ?	
21	4 R/S	PRED. ACT = ?	
22	0 R/S	ACTIVITY # = ?	
23	0 R/S	ANY CHANGES (Y/N)?	
24	N R/S	ENTER ENDING ACTIVITIES END ACT. # = ?	See Note 2
25	5 R/S	END ACT. # = ?	
26	0 R/S	START ACT. # = ?	See Note 3
27	1 R/S	ACT # = 1.	
28	R/S	DURATION = 15.	See Note 4
29	R/S	ES = 0.	
30	R/S	EF = 15.	
31	R/S	LS = 0.	
32	R/S	LF = 15.	
33	R/S	TF = 0	
34	R/S	ACT # = 2.	
35	R/S	DURATION = 20.	
36	R/S	PRED. ACT = 1.	
37	R/S	ES = 15.	
38	R/S	EF = 35.	
39	R/S	LS = 31.	
40	R/S	LF = 51.	
41	R/S	TF = 16.	
42	R/S	ACT # = 3.	
43	R/S	DURATION = 18.	
44	R/S	PRED. ACT = 1.	

Figure 5. CPM program problem example.

Step	Press	Resulting Display	Comments
45	R/S	ES = 15.	
46	R/S	EF = 33.	
47	R/S	LS = 15.	
48	R/S	LF = 33.	
49	R/S	TF = 0.	
50	R/S	ACT # = 4.	
51	R/S	DURATION = 18.	
52	R/S	PRED. ACT = 3.	
53	R/S	ES = 33.	
54	R/S	EF = 51.	
55	R/S	LS = 33.	
56	R/S	LF = 51.	
57	R/S	TF = 0.	
58	R/S	ACT # = 5.	
59	R/S	DURATION = 20.	
60	R/S	PRED. ACT = 2.	
61	R/S	PRED. ACT = 4.	
62	R/S	ES = 51.	
63	R/S	EF = 71.	
64	R/S	LS = 51.	
65	R/S	LF = 71.	
66	R/S	TF = 0.	
67	R/S	END PROGRAM	

Notes:

1. Activity numbers must be positive integers, beginning with 1. The numbers must be consecutive, but need not be entered sequentially.

2. Ending activities are those activities that do not precede any other activity.

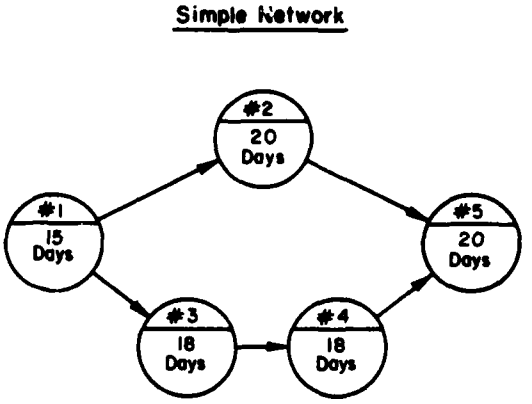
3. The starting activity number will tell the program where you want to begin to output the results.

4. If the printer is connected and in the NORM mode, lines 28 through 67 will be output automatically. You do not have to press the R/S key.

Figure 5. (Cont'd).

```
              RUN "CPM"
TOTAL #ACTIVITIES=?
              5.        RUN
ACTIVITY#=?
              1.        RUN
DURATION=?
             15.        RUN
PRED. ACT=?
              0.        RUN
ACTIVITY#=?
              2.        RUN
DURATION=?
             20.        RUN
PRED. ACT=?
              1.        RUN
PRED. ACT=?
              0.        RUN
ACTIVITY#=?
              3.        RUN
DURATION=?
             16.        RUN
PRED. ACT=?
              1.        RUN
PRED. ACT=?
              0.        RUN
ACTIVITY#=?
              4.        RUN
DURATION=?
             18.        RUN
PRED. ACT=?
              3.        RUN
PRED. ACT=?
              0.        RUN
ACTIVITY#=?
              5.        RUN
DURATION=?
             20.        RUN
PRED. ACT=?
              2.        RUN
PRED. ACT=?
              4.        RUN
PRED. ACT=?
              0.        RUN
ACTIVITY#=?
              0.        RUN
ANY CHANGES(Y/N)?
N                       RUN
 ENTER ENDING ACTIVITIES
END ACT.#=?
              5.        RUN
END ACT.#=?
              0.        RUN
START ACT.#=?
              1.        RUN
SEE KEY(Y/N)?
Y                       RUN
```

```
 ------------
/ES  ACT#  EF/
/   DURATION  /
/LS        LF/
 ------------
```



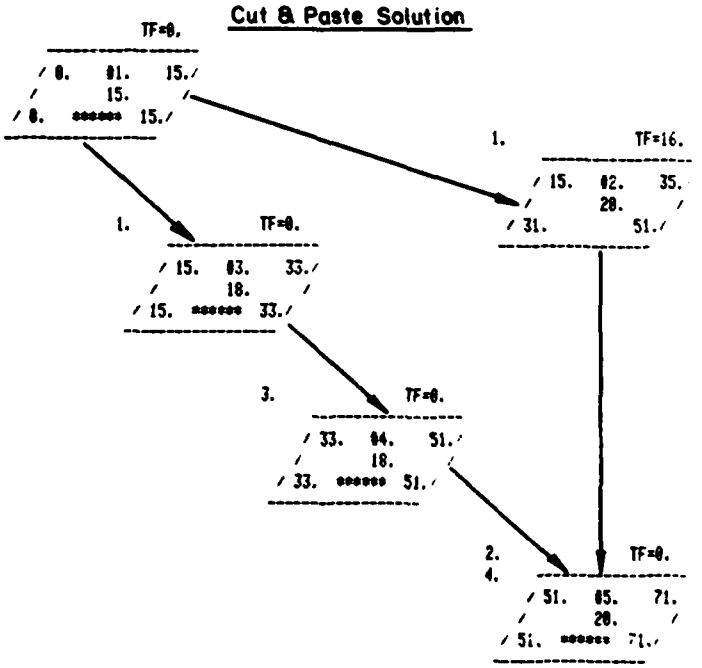


Figure 6. CPM program example (with printer).

4 THE ROAD CRATER PROGRAM (CRATER)

The road crater program computes the amount of explosive, number of cratering charges, and the number and depth of holes needed to produce hasty, deliberate, or relieved-face road craters for lengths of crater you specify.

General Program Information

Abbreviations used in the CRATER program are listed in Table 5. Input variable operating limits are listed in Table 6.

Program Sequence

The typical sequence of events and the options you encounter when executing this program are shown in Figure 7.

CRATER Program Example

Determine how much TNT, how many boreholes, and how many 40-lb cratering charges are required to blast a 41-ft-long deliberate crater.

Figure 8 shows how to solve this problem using the CRATER program. Figure 9 gives three examples of CRATER program output with printer attached.

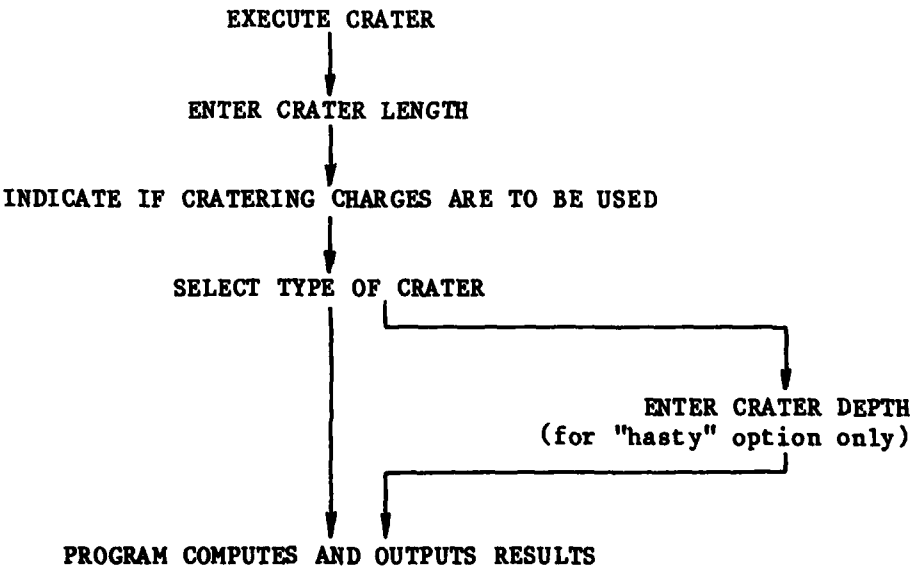


Figure 7. CRATER program sequence.

Table 5

CRATER Program Abbreviations

Symbol	Meaning
CHG	Charge
EXPLO	Explosive(s)
FT	Feet
LBS	Pounds
(Y/N)	(Yes/No)
#	Number
	Sum of (Total)

Table 6

CRATER Program Input Variable Operating Limits

Variable	Units	Minimum	Maximum
Crater Length	Feet	12	999
Crater Depth	Feet	7.5	15

Source Reference Notes:

FM 5-34 (September 1976), pp 28-32
FM 5-35 (February 1971), pp 3-21 through 3-25

Step	Press	Resulting Display	Comments
1	XEQ ALPHA CRATER ALPHA	CRATER LENGTH, (FT) = ?	
2	41 R/S	USE CRATER CHARGE (Y/N)?	
3	Y R/S	CRATER TYPE:	See Note 1
4	R/S	HASTY (Y/N)?	
5	N R/S	DELIBERATE (Y/N)?	
6	Y R/S	# 7 FT. HOLES = 4.	See Note 2
7	R/S	# 5 FT. HOLES = 2.	
8	R/S	# CRATER CHG = 10.	
9	R/S	PRIMER: TNT, LBS = 2.	
10	R/S	EXPLO, LB = 402.	
11	R/S	ALSO: NEED SHAPE CHARGES	
12	R/S	TO BLAST BOREHOLES!	
13	R/S	END PROGRAM	

Notes:

1. Crater-type Menu Order: Hasty, Deliberate, Relieved-Face

2. If a printer is connected and in the NORM mode, Steps 7 through 13 will be output automatically.

Figure 8. CRATER program problem example.

```
                XROM  "CRATER"
CRATER LENGTH,(FT)=?
              41.0       RUN
USE CRATER CHARGE(Y/N)?
N                        RUN
CRATER TYPE:
HASTY(Y/N)?
Y                        RUN
CRATER DEPTH,(FT)=?
              7.5        RUN

#HOLES=6.
HOLE DEPTH,FT=5.0
EXPLOSIVE, LBS/HOLE=50.

EXPLO,ΣLB=300.
ALSO:  NEED SHAPE CHARGES
TO BLAST BOREHOLES!
END PROGRAM
```

```
                XROM "CRATER"
CRATER LENGTH,(FT)=?
              41.0       RUN
USE CRATER CHARGE(Y/N)?
Y                        RUN
CRATER TYPE:
HASTY(Y/N)?
N                        RUN
DELIBERATE(Y/N)/?
Y                        RUN

#7FT.HOLES=4.
#5FT.HOLES=2.
CRATER CHG=10.
PRIMER:TNT,LBS=2.

EXPLO, ΣLB=402.
ALSO:  NEED SHAPE CHARGES
TO BLAST BOREHOLES!
END PROGRAM
```

```
                XROM "CRATER"
CRATER LENGTH,(FT)=?
              41.0       RUN
USE CRATER CHARGE(Y/N)?
Y                        RUN
CRATER TYPE:
HASTY(Y/N)?
N                        RUN
DELIBERATE(Y/N)?
N                        RUN
RELIEVED FACE(Y/N)?
Y                        RUN

FRIEND SIDE:
#5FT.HOLES=6.
#CRATER CHG=6.
PRIMER:TNT,LBS=6.

ENEMY SIDE:
#4FT.HOLES=5.
TNT,LBS=150.

EXPLO,ΣLB=396.
ALSO:  NEED SHAPE CHARGES
TO BLAST BOREHOLES!
END PROGRAM
```

Figure 9. CRATER program examples (with printer).

5 THE DEMOLITION PROGRAM (DEMO)

The demolition program determines the amount of explosive required, the number of explosive units, the number of charges, and the minimum safe distance for the following engineer activities: cutting timber, cutting steel, and breaching walls. A menu of explosive types to be used is also presented. The program has three timber cutting options: internal charge placement, external placement, and abatis. The steel cutting options cover four application areas: railroad rails, round steel sections, structural steel sections, and carbon steel rods. The breaching applications are used in conjunction with applicable tables in FM 5-34.

General Program Information

Abbreviations used in the DEMO program are listed in Table 7. Input variable operating limits are listed in Table 8.

Program Sequence

The typical sequence of events and the options you encounter when executing this program are shown in Figure 10.

DEMO Program Example

Determine the amount of explosive required, the number of explosive units, the number of charges, and the miminum safe distance required to breach

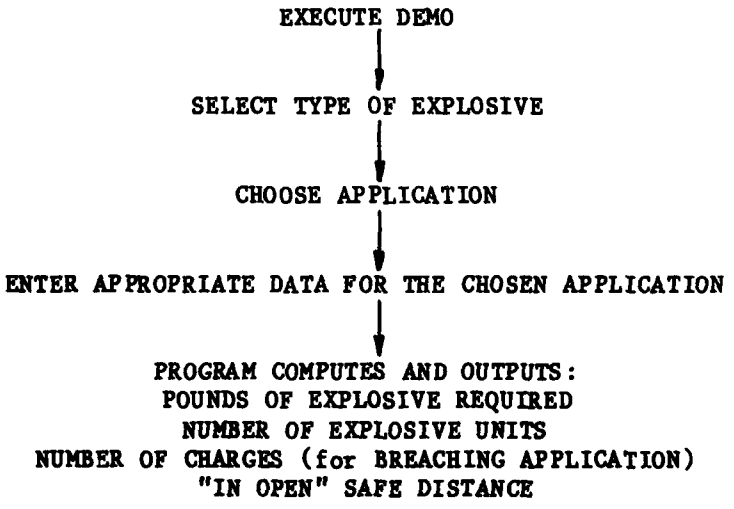


Figure 10. DEMO program sequence.

Table 7

DEMO Program Abbreviations

Symbol	Meaning
BREACH	Breaching
C	Tamping Factor
CONCR	Concrete
DEMO	Demolition
DIA	Diameter
DIST	Distance
EXPLO	Explosive
EXTERN	External
FT	Feet
HT	Height
IN	Inches
INTERN	Internal
K	Material Factor
LB	Pound(s)
M	Meter
MIN	Minimum
REINF	Reinforced
REQD	Required
RR	Railroad
SECT	Section
STL	Steel
STR.STL.	Structural Steel
SQ. IN	Square Inches
TAB	Table
X-SECT	Cross-Sectional
(Y/N)	(Yes/No)
#	Number

Table 8

DEMO Program Input Variable Operating Limits

Variable	Units	Minimum	Maximum
Timber Diameter	Inches	0.5	180
Rail Height	Inches	1	9
Bar Diameter	Inches	0	24
X-Sect Area	$(Inch)^2$	0	99
Section Diameter	Inches	0	99
Material Factor, K		0.07	1.76
Tamp Factor, C		1	3.6
Barrier Width	Feet	0	999
Breaching Radius	Feet	0.1	99

Source References:

FM 5-34 (September 1976), pp 24-29
FM 5-29 (February 1971), pp 1-6 and 3-20

a reinforced wall 7-1/2 ft thick and 42 ft long, using ground placed, tamped charges.

Figure 11 shows how to solve this problem using the DEMO program. Figure 12 gives three examples of DEMO program output with printer attached.

Step	Press	Resulting Display	Comments
1	XEQ		
	ALPHA		
	DEMO		
	ALPHA	EXPLOSIVE TYPE:	See Note 1.
2	R/S	TNT (Y/N)?	
3	N R/S	M112 C4 (1.25 LB) (Y/N)?	
4	N R/S	M5A1 C4 (2.5 LB) (Y/N)?	
5	N R/S	DYNAMITE, M1 (Y/N)?	
6	Y R/S	APPLICATION:	
7	R/S	CUT TIMBER (Y/N)?	
8	N R/S	CUT STEEL (Y/N)?	
9	N R/S	BREACH (Y/N)?	
10	Y R/S	TAB. 2-3, FM 5-34:	See Note 2.
11	R/S	MATERIAL FACTOR, K = ?	
12	.54 R/S	TAB. 2-4, FM 5-34:	See Note 3.
13	R/S	TAMP FACTOR, C = ?	
14	2 R/S	BARRIER WIDTH, (FT) = ?	
15	42 R/S	BREACH. RADIUS, (FT) = ?	
16	7.5 R/S	REQD. EXPLO, LBS = 1,486.5	
17	R/S	# EXPLO. UNITS = 2,973.	See Note 4.
18	R/S	# CHARGES = 3.	
19	R/S	OPEN, SAFE DIST, M = 1.141.	
20	R/S	END PROGRAM	

Notes:

1. Explosive Type Menu Order: TNT, M112 C4(1.25 lb), M5A1 C4(2.5 lb), Dynamite, Tetrytol, M118 Sheet (0.5 lb), M186 Roll (25 lb)

2. See Table 2-3, FM 5-34 to determine the material factor, K

3. See Table 2-4, FM 5-34 to determine the tamp factor, C

4. If a printer is connected and in the NORM mode, steps 17 through 20 will be output automatically.

Figure 11. DEMO program problem example.

```
              XROM  "DEMO"
EXPLOSIVE TYPE:
TNT(Y/N)?
N                          RUN
M112 C4(1.25LB)(Y/N)?
Y                          RUN
APPLICATION:
CUT TIMBER(Y/N)?
Y                          RUN
TIMBER DIA.(IN)=?
             24.0          RUN
CHARGE PLACEMENT:
ABATIS(Y/N)?
N                          RUN
EXTERN.(Y/N)?
Y                          RUN

REQD.EXPLO,LBS=11.3
#EXPLO.UNITS=9.
IN OPEN,SAFE DIST,M=300.
END PROGRAM
```

```
              XROM  "DEMO"
EXPLOSIVE TYPE:
TNT(Y/N)?
Y                          RUN
BLOCKS,TNT,1LB(Y/N)?
N                          RUN
APPLICATION:
CUT TIMBER(Y/N)?
N                          RUN
CUT STEEL(Y/N)?
Y                          RUN
TYPE STEEL:
RR.RAIL(Y/N)?
Y                          RUN
RAIL.HT.(IN)=?
             5.0           RUN
RR.FROG(Y/N)?
N                          RUN

REQD.EXPLO,LBS=1.0
#EXPLO,UNITS=2.
IN OPEN,SAFE DIST,M=300.
END PROGRAM
```

```
                XROM "DEMO"
EXPLOSIVE TYPE:
TNT(Y/N)?
N                          RUN
M112 C4(1.25LB)(Y/N)?
N                          RUN
M5A1 C4(2.5LB)(Y/N)?
N                          RUN
DYNAMITE,M1(Y/N)?
Y                          RUN
APPLICATION:
CUT TIMBER(Y/N)?
                           RUN
CUT STEEL(Y/N)?
N                          RUN
BREACH(Y/N)?
Y                          RUN
TAB.2-3,FM5-34:
MATERIAL FACTOR,K=?
              .54          RUN
TAB.2-4,FM5-34:
TAMP FACTOR,C=?
              2.0          RUN
BARRIER WIDTH,(FT)=?
             42.0          RUN
BREACH, RADIUS,(FT)=?
              7.5          RUN

REQD.EXPLO,LBS=1,486.5
#EXPLO.UNITS=2,973.
#CHARGES=3.
 OPEN,SAFE DIST,M=1,141.
END PROGRAM
```

Figure 12. DEMO program examples (with printer).

6 THE MINEFIELD PROGRAM (MINES)

The minefield program computes the logistical requirements for installing a standard pattern minefield given the field density, the irregular outer-edge cluster composition, the field length and depth, and conditions under which the work is to be done.

General Program Information

Abbreviations used in the MINES program are listed in Table 9. Input variable operating limits are listed in Table 10.

Program Sequence

The typical sequence of events and the options encountered when executing the MINES program is shown in Figure 13.

MINES Program Example

Determine the logistical requirements for a minefield, 400-m long and 400-m deep, with an AT-APF-APB mine density of 1-1-0, and an AT-APF-APB mine Irregular Outer Edge (IOE) cluster composition of 1-2-2.

Figure 14 shows how to solve this problem using the MINES program. Figure 15 gives two examples of MINES program output with printer attached.

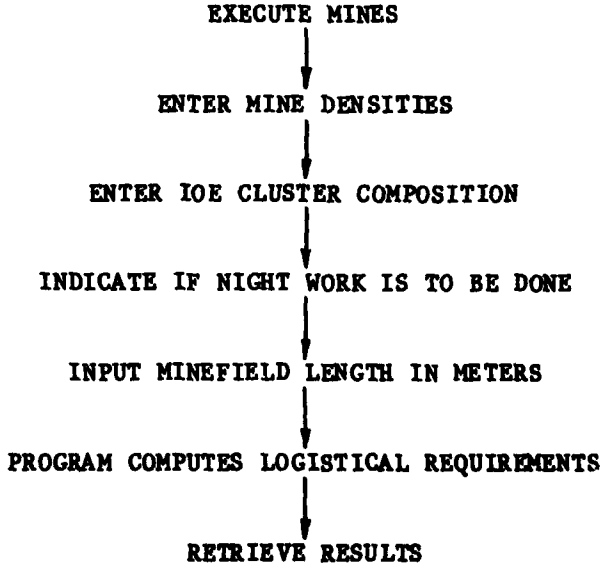


Figure 13. MINES program sequence.

Table 9

MINES Program Abbreviations

Symbol	Meaning
APB	Anti-Personnel Blast
APF	Anti-Personnel Fragmentation
AT	Anti-Tank
IOE	Irregular Outer Edge
M	Meter(s)
MAX	Maximum
MMF	Main Minefield
RL	Reel(s)
(Y/N)	Yes/No
#	Number

Table 10

MINES Program Input Variable Operating Limits

Variable	Units	Minimum	Maximum
Mine Densities:			
AT Mines/M	Mines/M	0	4
APF Mines/M	Mines/M	0	16
APB Mines/M	Mines/M	0	16
IOE Cluster Composition:			
No. of AT Mines	Each	0	1
No. of APF Mines	Each	0	5
No. of APB Mines	Each	0	5
Minefield Length	Meters	0	5000
Minefield Depth	Meters	0	999

Source Reference Notes:

FM 5-34 (September 1976), pp 54-59
FM 5-20 (November 1976)

Step	Press	Resulting Display	Comments
1	XEQ ALPHA MINES ALPHA	ENTER MINE DENSITY:	
2	R/S	# AT/M = ?	
3	3 R/S	# APF/M = ?	
4	4 R/S	# APB/M = ?	
5	8 R/S	IOE CLUSTER COMPOSITION:	
6	R/S	# AT = ?	
7	1 R/S	# APF = ?	
8	2 R/S	# APB = ?	
9	2 R/S	DO AT NIGHT (Y/N)?	
10	N R/S	FIELD LENGTH, (M) = ?	
11	400 R/S	FIELD DEPTH, (M) = ?	
12	400 R/S	TOTAL MINES:	
13	R/S	# AT = 1,370.	See Note 1.
14	R/S	# APF = 1,859.	
15	R/S	# APB = 3,619.	
16	R/S	IOE MINES:	
17	R/S	# AT = 45.	
18	R/S	# APF = 90.	
19	R/S	# APB = 90.	
20	R/S	MMF MINES:	
21	R/S	# AT = 1,200.	
22	R/S	# APF = 1,600.	
23	R/S	# APB = 3,200.	
24	R/S	# IOE CLUSTERS = 45.	
25	R/S	# STRIPS = 9.	
26	R/S	2-STRAND, 4-SIDE FENCE:	
27	R/S	# WIRE (RL) = 13.	
28	R/S	# SIGNS, PICKETS = 165.	
29	R/S	# SANDBAGS = 3,780.	
30	R/S	MANHOURS = 962.	
31	R/S	END PROGRAM	

Notes:

1. If a printer is connected and in the NORM mode, steps 13 through 31 will be output automatically.

Figure 14. MINES program problem example.

```
          XROM "MINES"
ENTER MINE DENSITY:
#AT/M=?
                1.00    RUN
#APF/M=?
                1.00    RUN
#APB/M=?
                2.00    RUN
IOE CLUSTER COMPOSITION:
#AT=?
                  1.    RUN
#APF=?
                  1.    RUN
#APB=?
                  1.    RUN
DO AT NIGHT(Y/N)?
Y                       RUN
FIELD LENGTH,(M)=?
              400.      RUN
FIELD DEPTH,(M)=?
              100.      RUN

TOTAL MINES:
#AT=490.
#APF=490.
#APB=930.
IOE MINES:
#AT=45.
#APF=45.
#APB=45.
MMF MINES:
#AT=400.
#APF=400.
#APB=800.
#IOE CLUSTERS=45.
#STRIPS=3.
2-STRAND,4-SIDE FENCE:
#WIRE(RL)=9.
#SIGNS,PICKETS=189.
#SANDBAGS=1,350.
MANHOURS=436.
END PROGRAM
```

```
          XROM "MINES"
ENTER MINE DENSITY:
#AT/M=?
                3.00    RUN
#APF/M=?
                4.00    RUN
#APB/M=?
                6.00    RUN
IOE CLUSTER COMPOSITION:
#AT=?
                  1.    RUN
#APF=?
                  2.    RUN
#APB=?
                  2.    RUN
DO AT NIGHT(Y/N)?
N                       RUN
FIELD LENGTH,(M)=?
              400.      RUN
FIELD DEPTH,(M)=?
              400.      RUN

TOTAL MINES:
#AT=1,370.
#APF=1,859.
#APB=3,619.
IOE MINES:
#AT=45.
#APF=90.
#APB=90.
MMF MINES:
#AT=1,200.
#APF=1,600.
#APB=3,200.
#IOE CLUSTERS=45.
#STRIPS=9.
2-STRAND,4-SIDE FENCE:
#WIRE(RL)=13.
#SIGNS,PICKETS=165.
#SANDBAGS=3,780.
MANHOURS=962.
END PROGRAM
```

Figure 15. MINES program examples (with printer).

7 THE WIRE OBSTABLE PROGRAM (WIRE)

The WIRE obstacle program computes the logistical requirements for installing any of seven common wire obstacles: double apron fence, 4- and 2-pace; double apron fence, 6- and 3-pace; high wire; low wire; 4-stand fence; triple standard concertina; and general purpose barbed tape obstacle. The program can also be used to compute the effective length of the obstacle according to its function and location on the battlefield. If you already know the effective length, you can input the effective length directly.

General Program Information

Abbreviations used in the WIRE program are listed in Table 11. Input variable operating limits are listed in Table 12.

Program Sequence

The typical sequence of events and the options encountered when executing this program are shown in Figure 16.

WIRE Program Example

Determine the effective length, the number of 300-m sections, the amount of material, the number of manhours, and the number of truckloads required to construct a triple-standard concertina obstacle. The entanglement is for protecting an area on the FEBA which has 100 m of actual front. Only one belt of wire is to be used. The construction is done at night with experienced troops.

Figure 17 shows how to solve this problem using the WIRE program. Figure 18 gives two examples of WIRE program output with printer attached.

Table 11

WIRE Program Abbreviations

Symbol	Meaning
DBL.	Double
EFF.LEN.	Effective Length
FEBA	Forward Edge of the Battle Area
GPBTO	General Purpose Barbed Tape Obstacle
M	Meter(s)
MED	Medium
T	Ton(s)
(Y/N)	(Yes/No)
3-STD	Triple standard
#	Number

Table 12

WIRE Program Input Variable Operating Limits

Variable	Units	Minimum	Maximum
Camp Perimeter	Meter	0	50,000
Length of Front	Meter	0	50,000
Unit Depth	Meter	0	5,000
Effective Length	Meter	0	2,250,000
Number of Belts	Each	1	9

Source Reference Notes:

FM 5-34 (September 1976), pp 105-109
FM 5-15 (June 1972), pp 6-10, 6-22, and 6-23

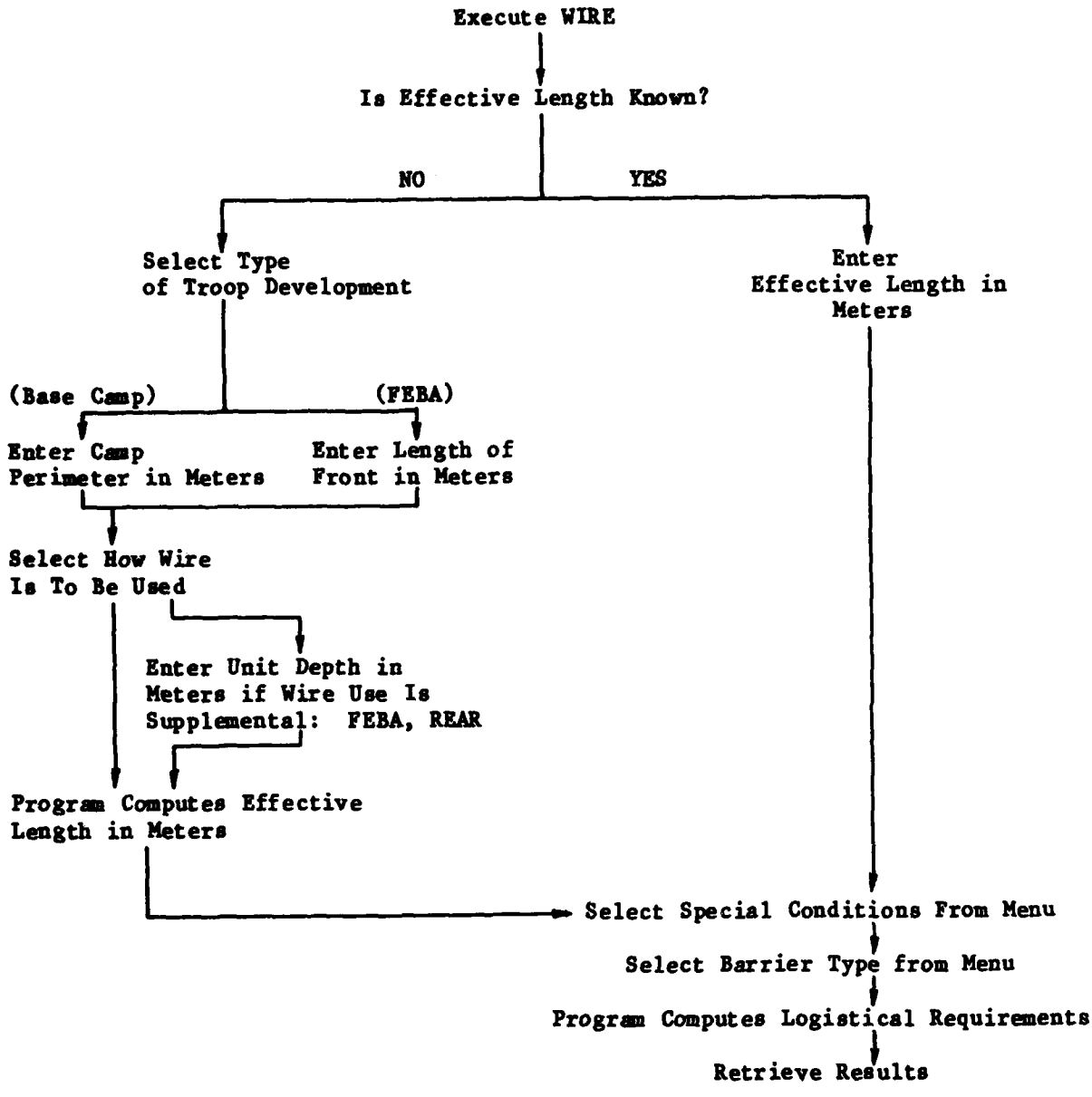


Figure 16. WIRE program sequence.

Step	Press	Resulting Display	Comments
1	XEQ ALPHA WIRE ALPHA	KNOW. EFF. LEN (Y/N)?	
2	N R/S	ON. FEBA (Y/N)?	
3	Y R/S	FRONT LENGTH, (M) = ?	
4	100 R/S	# BELTS = ?	
5	1 R/S	WIRE USE:	See Note 1.
6	R/S	TACTICAL (Y/N)?	
7	Y R/S	EFF. LEN, M = 125.	
8	R/S	USE BARBED TAPE (Y/N)?	
9	N R/S	EXPERIENCED TROOPS (Y/N)?	
10	Y R/S	USE DRIVEN PICKETS (Y/N)?	
11	N R/S	DO AT NIGHT (Y/N)?	
12	Y R/S	BARRIER TYPE:	See Note 2.
13	R/S	DBL APRON 4+2 (Y/N)?	
14	Y R/S	300M SECTIONS = 0.4	
15	R/S	PICKETS, LONG = 42.	See Note 3.
16	R/S	PICKETS, SHORT = 83.	
17	R/S	# WIRE REELS:	
18	R/S	USING U-PICKETS = 6.	
19	R/S	USING PICKET, SCREW = 6.	
20	R/S	USING PICKET, WOOD = 7.	
21	R/S	MANHOURS = 25.	
22	R/S	# 2.5^T LOADS = 0.3	
23	R/S	END PROGRAM	

Notes:

1. Wire Use Menu orders: Tactical, Protective, Supplemental

2. Barrier Type Menu Order: double apron, 4- and 2-pace; double apron, 6- and 3-pace; high wire; low wire; 4-wire fence; triple standard concertina; general purpose barbed tape obstacle

3. If a printer is connected and in the NORM mode, do not press the R/S key for Steps 15 through 23. The results will be output automatically.

Figure 17. WIRE program problem example.

```
            XROM "WIRE"
KNOW.EFF.LEN.(Y/N)?
N                   RUN
ON.FEBA(Y/N)?
Y                   RUN
FRONT LENGTH,(M)=?
            100.    RUN
#BELTS=?
              1.    RUN
WIRE USE:
TACTICAL(Y/N)?
Y                   RUN
EFF.LEN,M=125.
USE BARBED TAPE(Y/N)?
N                   RUN
EXPERIENCED TROOPS(Y/N)?
Y                   RUN
USE DRIVEN PICKETS(Y/N)?
N                   RUN
DO AT NIGHT(Y/N)?
Y                   RUN
BARRIER TYPE:
DBL APRON 4+2(Y/N)?
Y                   RUN

300M SECTIONS=0.4
PICKETS,LONG=42.
PICKETS, SHORT=83.
#WIRE REELS:
USING U-PICKETS=6.
USING PICKET,SCREW=6.
USING PICKET,WOOD=7.
MANHOURS=25.
#2.5'LOADS=0.3
END PROGRAM
```

```
            XROM "WIRE"
KNOW.EFF.LEN.(Y/N)?
Y                   RUN
EFF.LEN.(M)=?
          1,500.    RUN
USE BARBED TAPE(Y/N)?
N                   RUN
EXPERIENCED TROOPS(Y/N)?
N                   RUN
USE DRIVEN PICKETS(Y/N)?
Y                   RUN
DO AT NIGHT(Y/N)?
N                   RUN
BARRIER TYPE:
DBL APRON 4+2(Y/N)?
N                   RUN
DBL APRON 6+3(Y/N)?
N                   RUN
HIGH WIRE(Y/N)?
N                   RUN
LOW WIRE(Y/N)?
N                   RUN
4-WIRE FENCE(Y/N)?
N                   RUN
CONCERTINA,3-STD(Y/N)?
Y                   RUN

300M SECTIONS=5.0
PICKETS,LONG=800.
PICKETS, SHORT=20.
#WIRE REELS:
USING U-PICKETS=15.
USING PICKET,SCREW=15.
USING PICKET,WOOD=15.
ROLL,CONCERTINA=295.
STAPLES=1,585.
MANHOURS=150.
#2.5'LOADS=5.4
END PROGRAM
```

Figure 18. WIRE program examples (with printer).

8 GLOBAL UTILITY SUBROUTINES

The global subroutines are used by the six main application programs stored on MILENG1/UTIL and can be used by you to write your own programs. The subroutines can also be used by other military engineering programs that may one day be stored on ROMs designed to be used concurrently with the MILENG1/UTIL ROM. This convention will save a lot of room on future ROMs and in user-developed programs -- room that would otherwise be needed to store similar subroutines. If programmers adopt these subroutines, the military engineer community should find it easier to understand each others' programs.

Conventions

The following information on register use applies to all global utility subroutines. Registers 20 through 29 are reserved for existing and future global utility subroutines. Registers 20 through 23 are used to store temporarily up to 24 characters of a message that will be presented as a prompt of some sort to a program user. Each register stores up to six characters. Register 24 is an indirect storage register for a "pointer" to show where the next input value will be stored. Registers 25 and 26 store the minimum and maximum limits, respectively, of the current input variable. Registers 27 through 29 are reserved for global subroutines that may be developed in the future.

The following flag conventions are also used. Flag "10" records the results of a yes/no question. The flag is set if the response is "Y" (yes) and cleared if the response is "N" (no). Flag "9" is programmed into subroutine *I (numeric input) and *A (alpha input) to allow an answer already stored somewhere to be used instead of the user-defined values (alpha or numeric) that usually result when the *I and *A subroutines are invoked. Although this option is not used in any of the six main application programs on MILENG1/UTIL, the potential to bypass the normal user-input route and to use a value already in the system is available in these two subroutines. This flexibility may be useful in future applications.

Flag "8" was not used in the MILENG1/UTIL programs. It is reserved for future use as a "global use" flag, which "jumps over" certain parts of a program that do not have to be executed each time on repetitive runs. Flags "0" through "7" are reserved for application program use and are cleared each time the *F subroutine is invoked.

Global Subroutine *S

Running global subroutine *S should always be one of the first steps in any program; this insures that adequate data registers have been allocated for the program being executed. If enough registers are available, the subroutine returns to the main program and continues execution; if not, a user is prompted to resize the memory. The program must then be restarted.

Before using the subroutine, the entry condition must first be satisfied. The number of data registers required for the program is first loaded into the

X register. Note that this is the actual number required, not the number of the highest register, which would be one less because register 00 counts as one register. *S is then executed.

The following exit conditions are observed: *S returns to the calling program if adequate registers are available; otherwise, *S prompts to resize. Assume that the stack contents are destroyed upon the return.

A sample calling sequence follows:

Program Instruction	Explanation
61	Loads "61" into X register; says that 61 registers (0 through 60) are required for this particular program;
XEQ "*S"	Calls the *S subroutine
xxxxxxxxxx	Resume with main program

Note that *S also uses global subroutines *0 and *D.

Global Subroutine *F

This subroutine clears flags "00" through "07"; it should be used to initialize a program and to "clean up" at the end of each application program.

To use the subroutine, simply execute *F. *F returns to the calling program once flags "00" through "07" have been cleared.

A sample calling sequence follows:

Program Instruction	Explanation
XEQ *F	Calls the *F subroutine
xxxxx	Resume with main program

Global Subroutine *I

Global subroutine *I prompts for numeric input; a single tone will signal that input is required. If the input provided is not numeric, the prompt will be presented again.

The input value must pass a range check. If the input is out of range, a user is informed of the maximum or minimum acceptable value and reprompted. Global subroutine *I has a built-in option; if a user presses only the R/S key, he will be prompted with the current value.

To use *I, the indirect register pointer in register 24 must be set to the data register the input is to be stored in. This pointer is incremented

during each input call and has to be set only once if sequential input is to be stored in sequential registers. A prompt line and maximum and minimum acceptable value for the input are passed to the subroutine from the main program. The subroutine will not return to the main program until an acceptable value has been input. A "=?" is automatically appended to the prompt. If flag "9" is set, an "=" sign, the current value in the specified register, and a "?" are added to the prompt and displayed.

Entry conditions when calling *I are as follows: register X must contain the minimum acceptable value, register Y must contain the maximum acceptable value, register A must contain the prompt. Register 24 must contain the address of the register that the input is to be stored in.

The exit conditions from *I occur when an acceptable value has been entered; *I then returns to the calling program. (If flag "9" is set and R/S is pressed without numeric entry, the current value is used.) The input value is stored in the specified register and in the X register. The rest of the stack should be considered destroyed.

A sample calling sequence follows:

Program Instruction	Explanation
SF 09	Optional: used if "current value" of variable is to be presented to user for verification.
30	Identifies register address where input is to be stored.
STO 24	Stores indirect register address in register 24.
123	Specifies maximum acceptable input value
ENTER	Places maximum value in Y register
-37	Specifies minimum acceptable input value and puts in X register
"HEIGHT"	Specifies prompt to be presented to user
XEQ "*I"	Calls subroutine *I
CF 09	Used only if SF 09 option is used
xxxxxxxx	Resume with main program

Note that if input is sequential, the second two instructions only have to be programmed before the first variable is input. Also, if the current value option is used, make sure the calculator is FIX'd to the desired setting before calling *I. Note that *I also uses global subroutines *0 and *D.

Global Subroutine *0

This subroutine displays labeled output. A label is passed to the subroutine, and an "=" and the value in the X register are appended. A two-tone sequence signals output. The routine then displays the labeled answer. If a printer is attached, the output display is printed and the program continues execution after a pause. If no printer is attached, program execution stops until the the R/S key is pressed.

Before executing the subroutine, insure that register X contains the numeric data to be displayed and that register A contains the label to be appended. When the subroutine is done, it returns to the calling program. No registers are affected.

A sample calling sequence follows:

Program Instruction	Explanation
4.27	Puts value to be displayed in X register
"ANSWER"	Puts label to be used in alpha register
XEQ "*0"	Executes the *0 subroutine
xxxxxxxx	Resume with main program

Note that the calculator should be FIX'd to the desired accuracy before executing *0. Note that *0 also uses global subroutine *D.

Global Subroutine *D

This subroutine displays an alphanumeric text line. A two-tone sequence is used to signal the display; the routine then displays the contents of the alpha register. If a printer is attached, the output is printed and displayed; the program continues execution after a pause. If no printer is attached, program execution stops until the R/S key is pressed.

Before executing *D, insure that register A contains the alphanumeric text to be displayed. When *D is done, it returns to the calling program. No registers are affected.

An example calling sequence follows:

Program Instructions	Explanation
"SAMPLE"	Puts text to be displayed in alpha register
XEQ "*D"	Execute the *D subroutine
xxxxxxxx	Resume with main program

Global Subroutine *Y

This subroutine takes a prompt that is passed to it and appends "(Y/N)?" to it. A user must then respond with "Y" or "N" or the routine will reprompt. The answer to the query is returned to the calling program as flag "10" status. For "Y" responses, flag "10" is set; for "N" responses, it is cleared. The routine automatically places the calculator in the alpha mode before prompting and turns off the alpha mode after a response is input.

Before executing *Y, insure that register A contains the query text. When *Y is done, it returns to the calling program. Flag "10" status is affected.

A sample calling sequence follows:

Program Instructions	Explanation
"PRINT"	Puts query text in alpha register
XEQ "*Y"	Executes the *Y subroutine
FS? 10	Tests response: set=yes; clear=no
xxxxxxxx	Resume with main program

Global Subroutine *A

This subroutine prompts a user for alpha input. A maximum of 12 alpha characters are stored. A tone sounds to signal that input is required. A built-in option will allow you to prompt with the "current value" of the alpha variable. Under this option, if R/S is pressed, the "current value" of the text will be used when the program continues execution.

This subroutine requires that the indirect register address (pointer) stored in register 24 be set to the data register the alpha input is to be stored in. The alpha input is stored in two registers, six characters in each. The pointer in register 24 is incremented twice during each input call, so it only has to be set once if a string of input is to be put in sequential registers. A prompt line to label the input is passed to the subroutine, and a "=?" is added to this prompt by the subroutine.

To use the subroutine, the entry conditions must be satisfied. Register "A" must contain the prompt. Register 24 must contain the address of the register that the first six characters of the input will be stored in. If you set flag "9" before calling *A, the "current value" in the specified registers will be appended to the prompt and will be presented for verification. If a user agrees with the alpha value assigned, he would press the R/S key, and the program would use that alpha value for the variable. If a user elects to change the value, he would simply enter the correct alpha string (12 characters or fewer) and then press the R/S key. This new alpha value would then be used in place of the "current value."

The following exit conditions result: *A returns to the calling program, and the input value is stored in the specified registers. No other registers are affected.

A sample calling sequence follows:

Program Instructions	Explanation
SF 09	(Optional) If current value of variable is to be presented to user
30	First register address where input is to be stored
STO 24	Stores first indirect register address in register 24
"VARIABLENAME"	Specifies prompt to be presented to user
XEQ "*A"	Calls the "*A" subroutine
CF 09	Used only with "SF 09" option.
xxxxxxxx	Resume with main program.

Note that if input is sequential, only the second and third instructions need to be programmed before the first variable is input.

Global Subroutine *C

This subroutine clears a specified range of registers by storing a "0" in them. This subroutine should be used instead of CLRG so that the contents of registers not used in the application program are preserved.

Before the subroutine can be used, the entry conditions must be satisfied. Register X must contain the range of registers to be cleared; the format is fff.lll, where fff is the address of the first register to be cleared and lll is the address of the last register to be cleared.

After the specified registers have been cleared, *C exits to the calling program. The stack should be considered destroyed.

A sample calling sequence follows:

Program Instructions	Explanation
30.045	Specifies address of registers to be cleared (30 through 45)
XEQ "*C"	Calls the *C subroutine
xxxxxxxx	Resume with main program

Global Subroutine *R

This subroutine "rounds-up" a value and displays the integer portion of the number. The subroutine first adds 0.99 to the value stored in register X, then uses the integer portion of that value. Before entering the subroutine, insure that register X contains the value to be rounded. When *R is done, it returns to the calling program.

A sample calling sequence follows:

Program Instructions	Explanation
5.49	Specifies value to be rounded; could also be a recall RCL instruction
XEQ *R	Calls the *R subroutine
xxxxxxxx	Resume with main program

Global Subroutine *P

This subroutine displays the "END PROGRAM" message in large type. A two-tone sequence alerts the user; *P then displays the message. There are no pre-entry conditions for *P. The subroutine is called directly with the instruction, "XEQ *P". When *P is done, it returns to the calling program; no registers are affected.

CERL DISTRIBUTION

Chief of Engineers
ATTN: Tech Monitor
ATTN: DAEN-ASI-L (2)
ATTN: DAEN-ZCM

FESA, ATTN: Library 22060

416th Engineer Command 60623
ATTN: Facilities Engineer

ROK/US Combined Forces Command 96301
ATTN: EUSA-HHC-CFC/Engr

US Military Academy 10996
ATTN: Dept of Geography &
Computer Science

Engineer Studies Center 20315
ATTN: Library

AMMRC, ATTN: DRXMR-WE 02172

USA ARRCOM 61299
ATTN: DRCIS-RI-I
ATTN: DRCIS-IS

DLA, ATTN: DLA-WI 22314

FORSCOM
FORSCOM Engineer, ATTN: AFEN-FE

Fort Belvoir, VA 22060
ATTN: ATZA-DTE-EM
ATTN: ATZA-DTE-SW
ATTN: ATZA-FE
ATTN: Engineer Library
ATTN: Canadian Liaison Officer (2)
ATTN: IWR Library

Cold Regions Research Engr Lab 03755
ATTN: Library

ETL, ATTN: Library 22060

Waterways Experiment Station 39180
ATTN: Library

HQ, XVIII Airborne Corps & Ft Bragg 28307
ATTN: Library

Defense Technical Info Center 22314
ATTN: DDA (12)

US Government Printing 22304
Receiving Section/Depository Copies (2)

US Army, Commander
ATTN: 7 Engr Bde 09154
ATTN: 18 Engr Bde 09164
ATTN: 20 Engr Bde 28307
ATTN: 130 Engr Bde 09165
ATTN: 2 Engr Grp 96301
ATTN: 36 Engr Grp 31905
ATTN: 937 Engr Grp 66442
ATTN: 1 Engr Bn 66442
ATTN: 2 Engr Bn 96224
ATTN: 3 Engr Bn 31313
ATTN: 4 Engr Bn 80913
ATTN: 5 Engr Bn 65473
ATTN: 7 Engr Bn 71459
ATTN: 8 Engr Bn 76545
ATTN: 9 Engr Bn 09162
ATTN: 10 Engr Bn 09701
ATTN: 11 Engr Bn 22060
ATTN: 12 Engr Bn 09111
ATTN: 13 Engr Bn 93941
ATTN: 14 Engr Bn 93941
ATTN: 15 Engr Bn 98433
ATTN: 16 Engr Bn 09696
ATTN: 17 Engr Bn 76546
ATTN: 19 Engr Bn 40121
ATTN: 20 Engr Bn 42223
ATTN: 23 Engr Bn 09165
ATTN: 27 Engr Bn 28307
ATTN: 30 Engr Bn 22060
ATTN: 34 Engr Bn 66442
ATTN: 39 Engr Bn 01433
ATTN: 43 Engr Bn 31905
ATTN: 44 Engr Bn 96483
ATTN: 46 Engr Bn 36362
ATTN: 52 Engr Bn 80913
ATTN: 54 Engr Bn 09026
ATTN: 62 Engr Bn 76544
ATTN: 65 Engr Bn 96857
ATTN: 76 Engr Bn 20755
ATTN: 78 Engr Bn 09351
ATTN: 79 Engr Bn 09360
ATTN: 82 Engr Bn 09139
ATTN: 84 Engr Bn 96857
ATTN: 92 Engr Bn 31313
ATTN: 94 Engr Bn 09175
ATTN: 237 Engr Bn 09176
ATTN: 249 Engr Bn 09360
ATTN: 293 Engr Bn 09034
ATTN: 299 Engr Bn 73503
ATTN: 307 Engr Bn 28307
ATTN: 317 Engr Bn 09757
ATTN: 326 Engr Bn 42223
ATTN: 547 Engr Bn 09175
ATTN: 548 Engr Bn 28307
ATTN: 549 Engr Bn 28307
ATTN: 549 Engr Bn 09081
ATTN: 559 Engr Bn 09165
ATTN: 563 Engr Bn 09154
ATTN: 565 Engr Bn 09164
ATTN: 588 Engr Bn 71459
ATTN: 649 Engr Bn 09081
ATTN: 652 Engr Bn 96858
ATTN: 802 Engr Bn 96271
ATTN: 864 Engr Bn 98433

National Guard Bureau 20310
Installation Division

Deponai, John M.
User's manual for MILENG1/UTIL read only memory module of the combat engineer programmable hand-held calculator. -- Champaign, Ill : Construction Engineering Research Laboratory ; available from NTIS, 19f2.
45 p. (Technical report / Construction Engineering Research Laboratory ; P-136)

1. Military engineering. 2. Programmable calculators. 3. HP41c. I. Title. II. Series: Technical report (Construction Engineering Research Laboratory) ; P-136.